WE CLAIM:

1. A method for computing paths through a data network that includes a subnetwork which introduces a subset intransitivity constraint on allowable paths through the data network, the method comprising:

using an abstracted map of the network that includes network elements (NEs) and subnetwork elements (SNEs), with links between pairs of the NEs and the SNEs to construct a directed graph that compensates for the subset intransitivity; and

applying a routing algorithm to compute paths from a start node to the other nodes of the directed graph.

2. The method as claimed in claim 1 wherein the routing algorithm used to compute paths is Dijkstra's algorithm.

3. The method as claimed in claim 1 wherein the constructing the directed graph comprises:

creating a node in the directed graph to represent each NE in the abstracted network map;

creating in the graph one ingress node and one egress node to represent each SNE in the abstracted network map, and defining an edge in the directed graph from the ingress node to the egress node;

for each link between two SNEs in the same subnetwork, defining edges in the directed graph from an ingress node representing a first SNE of the pair to the an egress node representing a second SNE of the pair, and from the ingress node

representing the second SNE of the pair to the egress node representing the first SNE of the pair;

for each link between a SNE and a NE, defining an edge from the egress node representing the SNE to a node representing the NE, and an edge from the node representing the NE to the ingress node representing the SNE; and

for each link between two NEs, defining two oppositely directed edges between the nodes representing the NEs.

4. The method as claimed in claim 3 further comprising:

assigning an effectively unlimited capacity to the edge in the directed graph between the ingress node and the egress node that represent each SNE.

5. The method as claimed in claim 3 further comprising:

receiving available capacity of each of the links in the abstracted network map; and

removing edges representing links that do not have capacity to support a desired path.

6. The method as claimed in claim 4 further comprising:

receiving a cost associated with each of the links in the abstracted network map;

assigning weights to edges of the directed graph, based on the cost associated with the respective links that the edges in the directed graph represent; and

assigning a null cost to the edge between the ingress and egress node that represents each SNE.

7. The method as claimed in claim 6 wherein the abstracted network map further comprises a second subnetwork which introduces a subset intransitivity constraint on allowable paths through the data network and the method further comprises:

defining, for each link in the network map between SNEs of the respective subnetworks, directed edges from the respective egress nodes to the respective ingress nodes that represent the SNEs.

8. The method as claimed in claim 7 wherein the abstracted network map comprises a full-mesh connected virtual subnetwork abstracted from at least a part of an optical subnetwork having a ring topology with edge networks connected to the SNEs of the virtual subnetwork.

9. The method as claimed in claim 7 wherein the abstracted network map comprises a full-mesh connected virtual subnetwork abstracted from at least a part of a passive optical subnetwork with edge networks connected to the SNEs of the virtual subnetwork.

10. A network element (NE) having a routing processor for computing paths through a data network that includes a subnetwork that introduces subset intransitivity constraints on allowable paths through the network, the network element comprising:

a memory for storing an abstracted network map that includes network elements (NEs), subnetwork elements (SNEs) and links between pairs of SNEs, and NEs; and

program instructions for constructing a directed graph that compensates for the subset intransitivity, and for applying a routing algorithm to compute paths from a node representing the NE to the other nodes of the graph.

11. The network element as claimed in claim 10 wherein the links are bi-directional network links, and a unidirectional path computed through the directed graph is uniquely associated with an allowable bi-directional route through the abstracted network map.

12. The network element as claimed in claim 10 wherein the program instructions comprise modules for:

creating a node for representing each NE in the abstracted network map;

creating one ingress node and one egress node for representing each SNE in the abstracted network map, and defining a directed edge from the ingress node to the egress node;

defining edges from respective ingress nodes to respective egress nodes for each link between two SNEs in the same subnetwork of the abstracted network map;

defining a directed edge from an egress node representing each SNE to a node representing any NE having a link to the SNE in the abstracted

network map, and a directed edge from the node representing the NE to an ingress node representing the SNE; and

defining two oppositely directed edges for each link between nodes representing NEs in the abstracted network map.

13. The network element as claimed in claim 12 further comprising:

program instructions for receiving available capacity information regarding each of the links in the abstracted network map;

program instructions for removing edges from the directed graph that represent links that do not have a predetermined available capacity; and

program instructions for assigning an unlimited capacity to the directed edge between the ingress and egress node representing each SNE.

14. The network element as claimed in claim 13 wherein the program instructions further comprise:

program instructions for receiving costs of each of the links in the network map;

program instructions for assigning weights to edges of the directed graph, the weights corresponding to costs of links that the edges represent; and

program instructions for assigning a null cost to links between the ingress node and the egress node that represent each SNE.

15. The network element as claimed in claim 14 wherein the abstracted network map further comprises a second

subnetwork that introduces an intransitivity constraint in the data network, further comprising program instructions for defining directed edges from egress nodes to ingress nodes for each link in the abstracted network map between gateway SNEs of the respective subnetworks.

16. The network element as claimed in claim 15 wherein the network routing processor is a component of another NE, which is a root node of the directed graph in the application of Dijkstra's algorithm, so that the paths are computed by the other NE.

17. A method for representing an abstracted network map of a data network as a directed graph to compensate for a subset intransitivity constraint introduced by a subnetwork of subnetwork elements on allowable paths through the data network, the abstracted network map including the subnetwork elements (SNEs) and other network elements (NEs) interconnected by links, the method comprising:

creating a node in the directed graph to represent each of the NEs in the abstracted network map;

creating one ingress node and one egress node to represent each SNE in the abstracted network map, and defining an edge from the ingress node to the egress node;

for each link in the abstracted network map between two SNEs in the subnetwork, defining edges from the ingress nodes to the egress nodes representing the respective SNEs interconnected by the link;

for each link in the network map interconnecting a SNE and a NE, defining an edge from the egress node representing the SNE to the node representing the NE, and an edge from the node representing the NE to the ingress node representing the SNE; and

for each link in the network map between two NEs, defining two oppositely directed edges between the nodes representing the two NEs.

18. The method as claimed in claim 17 wherein the abstracted network map comprises a plurality of subnetworks, each introducing respective subset intransitivity constraints on allowable paths through the network, and the method further comprises defining a directed edge from the egress node to the ingress node representing the respective gateway SNEs interconnected by a link.

19. The method as claimed in claim 18 further comprising assigning a weight to each edge in the graph in correspondence with a cost of using a respective bidirectional link in the network map.

20. The method as claimed in claim 19 further comprising assigning a null cost to each directed edge between the ingress and the egress node that represents the SNE.

21. The method as claimed in claim 20 further comprising removing all directed edges in the graph that represent links that do not have a predefined available capacity.

22. A computer product comprising a machine readable data signal containing executable instructions for performing the method claimed in claim 15.